**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NHA TRANG**



**BÁO CÁO THỰC TẬP CƠ SỞ**

**ĐỀ TÀI: VIẾT CHƯƠNG TRÌNH MINH HOẠ GIẢI THUẬT KRUSKALS TÌM CÂY KHUNG NHỎ NHẤT TRONG ĐỒ THỊ**

**Giáo viên hướng dẫn : Hà Thị Thanh Ngà**

**Sinh viên thực hiện : Trần Văn Khánh**

**MSSV : 61133801**

**Lớp : 61CNTT2**

# PHẦN 1: TỔNG QUAN VỀ ĐỀ TÀI NGUYÊN CỨU

## 1.1 CƠ SỞ LÝ THUYẾT

### Cây khung nhỏ nhất

Theo lý thuyết đồ thị, chúng ta đều biết rằng 1 đồ thị được biểu diễn bằng công thức G=(V,E), trong đó đồ thị G của chúng ta bao gồm tập các đỉnh V và tập các cạnh E.

* Cây khung của một đồ thị vô hướng liên thông 𝐺 là một cây chứa tất cả các đỉnh của 𝐺.
* Trong đồ thị có trọng số, cây khung nhỏ nhất (minimum spanning tree) là cây khung có tổng trọng số các cạnh trong cây nhỏ nhất.

**Tính chất:** Một vài tính chất cây khung nhỏ nhất trong đồ thị vô hướng có trọng số

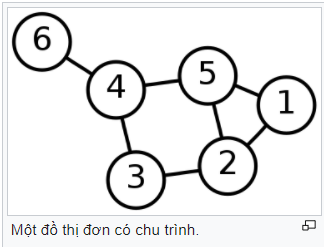
* **Tính chất chu trình:** Trong một chu trình C bất kỳ, nếu e là cạnh có trọng số lớn nhất tuyệt đối (không có cạnh nào có trọng số bằng e) thì e không thể nằm trên bất kỳ cây khung nhỏ nhất nào.
* **Đường đi hẹp nhất:** Xét 2 đỉnh u, v bất kỳ trong đồ thị. Nếu w là trọng số của cạnh lớn nhất trên đường đi từ u đến v trên cây khung nhỏ nhất của đồ thị thì ta không thể tìm được đường đi nào từ u đến v trên đồ thị ban đầu chỉ đi qua những cạnh có trọng số nhỏ hơn w.
* **Tính duy nhất:** Nếu tất cả các cạnh đều có trọng số khác nhau thì chỉ có duy một cây khung nhỏ nhất. Ngược lại, nếu một vài cạnh có trọng số giống nhau thì có thể có nhiều hơn một cây khung nhỏ nhất.
* **Tính chất cạnh nhỏ nhất:** Nếu e là cạnh có trọng số nhỏ nhất của đồ thị, và không có cạnh nào có trọng số bằng e thì e nằm trong mọi cây khung nhỏ nhất của đồ thị.

### Chu trình

Trong lý thuyết đồ thị, chu trình trong đồ thị là một dây chuyền đóng. Đồ thị chỉ gồm một chu trình với n đỉnh được gọi là đồ thị vòng, ký hiệu Cn

Các loại chu trình:

* Chu trình chẵn: là chu trình có độ dài chẵn.
* Chu trình lẻ: là chu trình có độ dài lẻ.
* Chu trình có hướng: là một chu trình đơn mà mọi cung trong đó đều cùng hướng, nghĩa là mọi đỉnh đều có bậc trong và bậc ngoài bằng 1. Có thể gọi đơn giản là chu trình khi ngữ cảnh rõ ràng.
* Chu trình đơn: là chu trình không đi qua một cạnh nào quá một lần.
* Chu trình sơ cấp: là chu trình không chứa cùng một đỉnh quá một lần (trừ đỉnh đầu và đỉnh cuối). Trong đồ thị ở hình trên, (1, 5, 2, 1) là một chu trình sơ cấp. Chu trình sơ cấp thì là chu trình đơn.
* Chu trình Euler: là chu trình qua tất cả các cạnh, mỗi cạnh đúng một lần.
* Chu trình bao trùm: là cách gọi khác của chu trình Hamilton.



**Hình 1.1 Minh hoà chu trình của một đồ thị**

### Cây

Cây là một đồ thị mà trong đó hai đỉnh bất kì đều được nối với nhau bằng đúng một đường đi. Nói cách khác, đồ thị liên thông bất kỳ không có chu trình là một cây. Rừng là hợp (disjoint union) của các cây. Cây được sử dụng rộng rãi trong các cấu trúc dữ liệu của ngành khoa học máy tính như cây nhị phân, đống, trie, cây Huffman cho nén dữ liệu, v.v...

Các điều kiện đầy đủ để đồ thị là một cây:

Cho đồ thị G=(V,E) có n đỉnh. Sáu mệnh đề sau là tương đương:

* G là một cây;
* G không có chu trình và có n-1 cạnh;
* G liên thông và có n-1 cạnh;
* G không có chu trình và nếu bổ sung vào một cạnh nối hai đỉnh không kề nhau thì xuất hiện một chu trình duy nhất;
* G liên thông và nếu bỏ đi một cạnh bất kỳ thì G mất tính liên thông;
* Mỗi cặp đỉnh trong G được nối với nhau bằng đường đi duy nhất.

### Thuật toán Kruskal

Thuật toán Kruskal là một thuật toán trong lý thuyết đồ thị để tìm cây bao trùm nhỏ nhất của một đồ thị liên thông có trọng số. Nói cách khác, nó tìm một tập hợp các cạnh tạo thành một cây chứa tất cả các đỉnh của đồ thị và có tổng trọng số các cạnh là nhỏ nhất.Thuật toán Kruskal là một ví dụ của thuật toán tham lam.

Thuật toán này xuất bản lần đầu tiên năm 1956, bởi Joseph Kruskal.

## 1.2 CÔNG CỤ THỰC HIỆN VÀ NGÔN NGỮ LẬP TRÌNH SỬ DỤNG

### TỔNG QUAN VỀ DEV C++

Dev-C++ là một môi trường phát triển tích hợp tự do (IDE) được phân phối dưới hình thức giấy phép Công cộng GNU hỗ trợ việc lập trình bằng C/C++. Nó cũng nằm trong bộ trình dịch mã nguồn mở MinGW. Chương trình IDE này được viết bằng ngôn ngữ Delphi. Dự án phát triển Dev-C++ được lưu trữ trên SourceForge. Dev-C++ nguyên được phát triển bởi một lập trình viên có tên là Colin Laplace và chỉ chạy trên hệ điều hành Microsoft Windows.

### NGÔN NGỮ LẬP TRÌNH C++

C++ là một loại ngôn ngữ lập trình bậc trung (middle-level). Đây là ngôn ngữ lập trình đa năng được tạo ra bởi Bjarne Stroustrup như một phần mở rộng của ngôn ngữ lập trình C, hoặc "C với các lớp Class", Ngôn ngữ đã được mở rộng đáng kể theo thời gian và C ++ hiện đại có các tính năng: lập trình tổng quát, lập trình hướng đối tượng, lập trình thủ tục, ngôn ngữ đa mẫu hình tự do có kiểu tĩnh, dữ liệu trừu tượng, và lập trình đa hình, ngoài ra còn có thêm các tính năng, công cụ để thao tác với bộ nhớ cấp thấp.

# PHẦN 2. ĐẶC TẢ BÀI TOÁN

## 2.1 BÀI TOÁN

Viết một chương trình minh họa giải thuật Kruskals tìm cây khung nhỏ nhất trong đồ thị.

Mục đích: Thể hiện giải thuật dưới dạng chương trình hoàn chỉnh.

Yêu cầu:

Mức 1:

1. Trình bày giải thuật thực hiện chương trình

2. Sử dụng ngôn ngữ: Pascal/C/C++ để thể hiện

3. Yêu cầu:

- Sử dụng cấu trúc để lưu cạnh của đồ thị

- Nhập số đỉnh, số cạnh, danh sách các cạnh của đồ thị

- Hiển thị kết quả cây khung nhỏ nhất.

**Phát biểu bài toán tìm cây khung nhỏ nhất:**

Cho đồ thị vô hướng, liên thông G(V,E) trong đó mỗi cạnh e∈E được gán một trọng số w(e)∈R+. Một cây khung là một đồ thị con của G, không có chu trình, và chứa mọi đỉnh của G. Trọng số của một cây khung là tổng trọng số các cạnh của cây khung đó. Tìm cây khung có trọng số nhỏ nhất của G(V,E).

## 2.2 **HƯỚNG DẪN XỬ LÝ BÀI TOÁN**

Sử dụng thuật toán Karuskal để giải quyết bài toán.

### Tư tưởng thuật toán

Thuật toán Kruskal dựa trên mô hình xây dựng cây khung nhỏ nhất bằng thuật toán hợp nhất.

* Thuật toán không xét các cạnh với thứ tự tuỳ ý.
* Thuật toán xét các cạnh theo thứ tự đã sắp xếp theo trọng số.

Để xây dựng tập n-1 cạnh của cây khung nhỏ nhất - tạm gọi là tập K, Kruskal đề nghị cách kết nạp lần lượt các cạnh vào tập đó theo nguyên tắc như sau:

* Ưu tiên các cạnh có trọng số nhỏ hơn.
* Kết nạp cạnh khi nó không tạo chu trình với tập cạnh đã kết nạp trước đó

### Mô tả thuật toán

Giả sử ta cần tìm cây bao trùm nhỏ nhất của đồ thị G. Thuật toán bao gồm các bước sau.

* Khởi tạo rừng F (tập hợp các cây), trong đó mỗi đỉnh của G tạo thành một cây riêng biệt
* Khởi tạo tập S chứa tất cả các cạnh của G
* Chừng nào S còn khác rỗng và F gồm hơn một cây
  + Xóa cạnh nhỏ nhất trong S
  + Nếu cạnh đó nối hai cây khác nhau trong F, thì thêm nó vào F và hợp hai cây kề với nó làm một
  + Nếu không thì loại bỏ cạnh đó.

Khi thuật toán kết thúc, rừng chỉ gồm đúng một cây và đó là một cây bao trùm nhỏ nhất của đồ thị G.

### Các bước thực hiện

Cho đồ thị G=(X,E).

Bước 1: Sắp xếp các cạnh của đồ thị theo thứ tự trọng số tăng dần.

Bước 2: Khởi tạo T:= Ø

Bước 3: Lần lượt lấy từng cạnh thuộc danh sách đã sắp xếp. Nếu T+{e} không chứa chu trình thì gán T:=T+{e}.

Bước 4: Nếu T đủ n-1 phần tử thì dừng, ngược lại làm tiếp bước 3.

### Mã giả

Procedure Kruskal;

𝑇:=∅;

For i:=1 To n-1 Do Begin

Chọn e là cạnh có trọng số nhỏ nhất trong E;

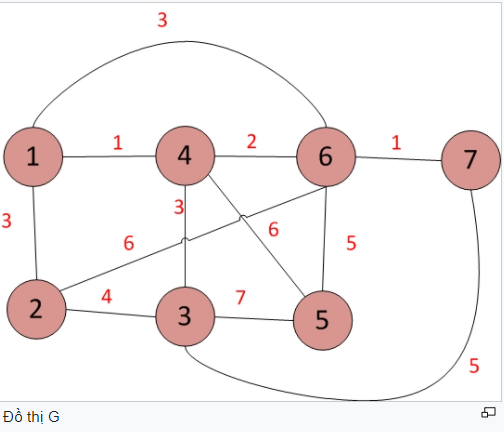
If (T ∪ {e} không chứa chu trình) Then T:= ∪ {e};

E:=E\{e};

End

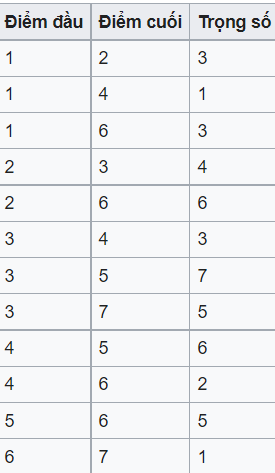
### Bài toán ví dụ minh hoạ

Cho đồ thị G như hình vẽ: Yêu cầu tìm ra cây khung nhỏ nhất của đồ thị



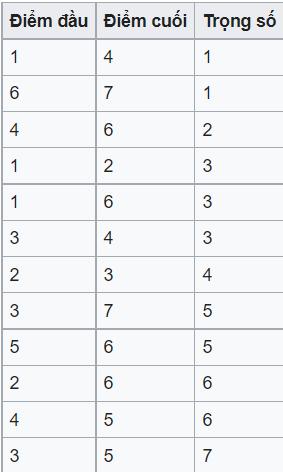
**Hình 2.1 Đồ thị bài toán minh hoạ**

* G gồm có 7 đỉnh
* Đồ thị G có n phần tử. Thuật toán Kruskal sẽ dừng khi có n-1 trong tập hợp T:
  + n = 7
  + Vậy số cạnh trong tập hợp T: n - 1 = 7 - 1 = 6(\*)
* **Bước 1:** Liệt kê tất cả cạnh với trọng số của cạnh đó: Dựa vào đồ thị ta liệt kê ra các cạnh gồm đỉnh đầu, đỉnh cuối và trọng số:

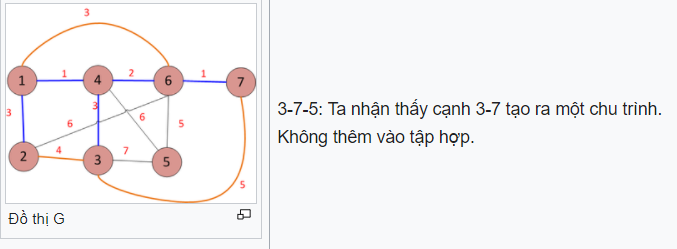
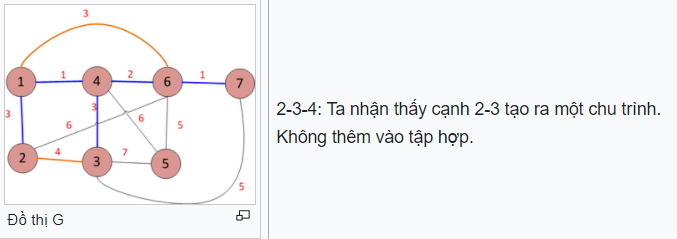
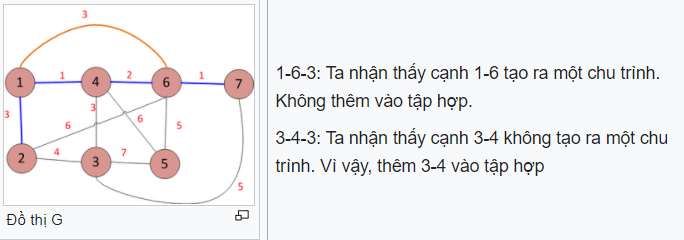
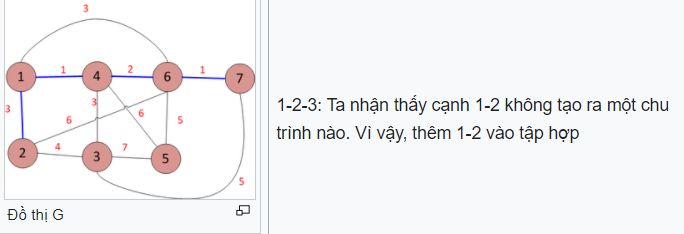
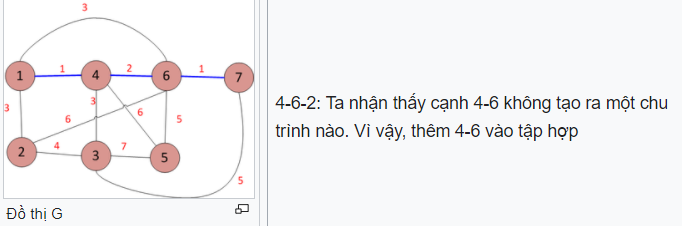
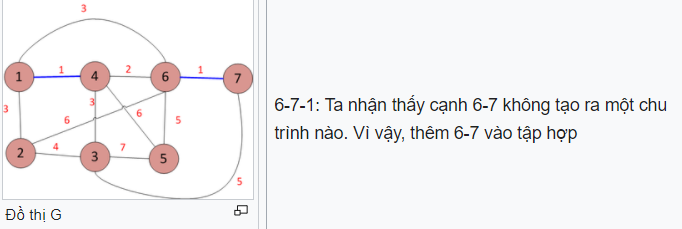
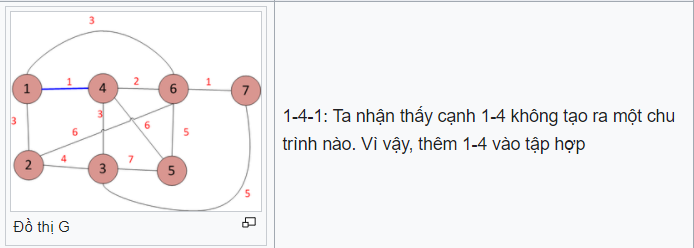


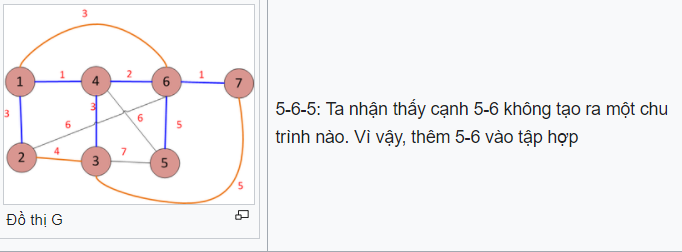
**Hình 2.2 Ma trận các cạnh và trọng số của nó**

* **Bước 2:** Sắp xếp các cạnh theo trọng số tăng dần:



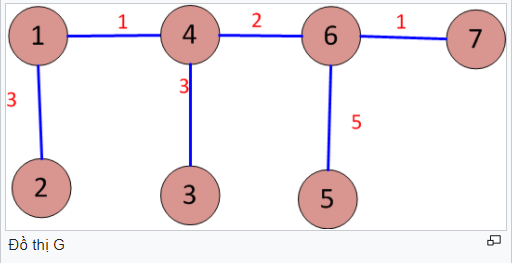
**Hình 2.3 Ma trận các cạnh và trọng số của nó đã sắp xếp**

**Bước 3:** Dựa vào kết quả ở bước 2. Ta tiến hành tìm cây khung bằng thuật toán Kruskal:  




Đến đây, ta đã tìm được 6 cạnh. Vậy kết thúc thuật toán. (Thỏa (\*))

Kết quả: Ta được đồ thị sau



**Hình 2.4 Kết quả cây khung nhỏ nhất cần tìm**

Với tổng chi phí là: Ta cộng tất cả các trọng số giữa các đỉnh lại với nhau

Vậy tổng chi phí: 3 + 1 + 3 + 2 + 5 + 1 = 15.

# PHẦN 3 CÀI ĐẶT CHƯƠNG TRÌNH

## 3.1 CODE CHƯƠNG TRÌNH

### Khai báo biến và thư viện

Khai báo thư viện C++ và namespace:

* Thư viện xử lý file: <fstream>
* Thư viện xử lý chuỗi: <string.h>

#include <fstream>

#include <string.h>

**using namespace** std;

Khai báo các biến mặc đinh và biến toàn cục:

#define fi "KRUSKAL.INP" // ten file nhap

#define fo "KRUSKAL.OUT" /ten /file xuat

**const int** MAXN = 1000;

**const int** MAXM = 1000;

**int** n, m; // n so dinh, m so canh

Canh a[MAXM];

**int** father[MAXN];

fstream f;

Khai báo cấu trúc 1 cạnh:

**struct** Canh {

**int** dinhdau, dinhcuoi;

**int** dodai;

};

### Xây dựng hàm

Xây dựng hàm **Nhap()** có chức năng nhập dữ liệu đâu vào:

Đầu tiên sử dụng phương thức *open()* của biến *f* truyền vào tham số đầu là đại chỉ file nhập, tham số thứ 2 có chức năng mở file để đọc. Đọc lần lược các biến n và m ở dòng đầu tiên(trong đó n là số đỉnh, m là số cạnh), sau đó chạy vòng lập for để đọc các dòng tiếp theo mỗi dòng trong file sẽ đọc từ số đầu tiên là *dinhdau*, số thứ 2 là *dinhcuoi*, số thứ 3 là *dodai*. Kết thúc vòng for() sử dụng phương thức *close()* để đóng file.

**void** Nhap() {

f.open(fi, ios::in);

f >> n >> m;

for (int i=1; i <= m; i++) {

f >> a[i].dinhdau >> a[i].dinhcuoi >> a[i].dodai;

}

f.close();

}

Xây dựng hàm **SapXep()**  có chức năng sắp xếp lại các cạnh theo thứ tự độ dài tăng dần.

Đầu tiên khai báo biến *tg* có kiểu dữ liệu struct là *Canh* để làm biến trung gian khi sắp xếp*,* chạy vòng lập *for()* để sắp xếp lại các cạnh theo thứ tự tăng dần theo độ dài. Thuật toán sắp xếp là bubbleSort hay còn gọi là thuật toán nổi bọt.

**void** SapXep() {

Canh tg;

for (**int** i=1; i<=m-1; i++)

for (**int** j=i+1; j <= m; j++)

if (a[i].dodai > a[j].dodai) {

tg = a[i];

a[i] = a[j];

a[j] = tg;

}

}

Xây dựng hàm **root()** có chức năng tìm nút cha của một đỉnh.

**int** root(**int** x) {

**if** (father[x] == -1) **return** x;

**else** **return** root(father[x]);

}

Xây dựng hàm **hopnhat()** có chức năng nối 2 nút lại với nhau.

**void** hopnhat(**int** r1, **int** r2) { /// r1 khac r2

**if** (r1 < r2) father[r2] = r1;

**else** /// r1 > r2

father[r1] = r2;

}

Xây dựng hàm **Kruskal()** đây là hàm sử dụng thuật toán Kruskal để tìm cây khung nhỏ nhất của độ thị

Đầu tiên ta gọi lại hàm **SapXep()** để sắp xếp lại các cạnh theo thứ tự tăng dần của độ dài. Sau đó ta khai báo các nút cha đều bằng vòng lập *for()*, khai báo các biến trung gian dem, tong, x, y, r1, r2 để thực hiện thuật toán Kruskal. Mở file *f* bằng phương thức *open()* truyền vào tham số thứ nhất là địa chỉ file xuất, tham số thứ 2 là *ios::out*  mở file để ghi kết quả của bài toán.

Chạy vòng lập for() từ dòng 1 đến hết số dòng trong file nhập, mỗi dòng sẽ thực hiện các thao tác sau:

* Nếu dem == số cạnh – 1 thì dừng lại thoát khỏi vòng lập
* Ngược lại gán x = a[i].dinhdau là phần tử thứ i của a có kiểu dữ liệu struct là Canh, gán y = a[i].dinhcuoi; gán r1 là nút cha của đỉnh x để tìm nút cha của đỉnh x, r2 là nút cha của đỉnh y để tìm nút cha của y.
* Nếu r1 = r2 thì r1 và r2 có cùng đỉnh gốc(cha)
* Vì vậy cần kiểm tra r1 và r2 không cùng 1 gốc(cha)
* Nếu r1#r2 ghi lại r1 và r2 vào file xuất và biến *dem* tăng lên 1, biển tong = tong (trước đó) + a[i].dodai để tính tổng khung nhỏ nhất sau đó kết nối đỉnh r1 và r2 lưu lại khung bao trùm nhỏ nhất.

Kết thúc vòng lập *for()* thì in ra tổng khung nhỏ nhất của đồ thị và đòng file f bằng phương thức *close().*

**void** Kruskal() {

SapXep();

**for** ( **int** i=1; i <= n; i++) father[i] = -1;

**int** dem = 0;

**int** tong = 0;

**int** x, y;

**int** r1, r2;

f.open(fo, ios::out);

**for** (i=1; i <= m; i++) {

**if** (dem == n-1) **break**;

x = a[i].dinhdau;

y = a[i].dinhcuoi;

r1 = root(x);

r2 = root(y);

**if** (r1 != r2) {

f << x << " " << y << endl; /// end line

dem++;

tong = tong + a[i].dodai;

hopnhat(r1, r2);

}

}

f << "Tong trong so cua cay khung be nhat la: " << tong;

f.close();

}

Hàm **main()** gọi lại hàm **Nhap() và** hàm **Kruskal()**  để chạy chương trình.